Authentication and Data Integrity

Cryptography

School of Engineering and Technology
CQUniversity Australia

Prepared by Steven Gordon on 23 Dec 2021,
auth.tex, r1951

Contents

Attacks on Information Transfer

1. Disclosure: encryption
2. Traffic analysis: encryption
3. Masquerade: message authentication
4. Content modification: message authentication
5. Sequence modification: message authentication
6. Timing modification: message authentication
7. Source repudiation: digital signatures
8. Destination repudiation: digital signatures

We have cover encryption primarily from the perspective of preventing disclosure attacks, i.e. providing confidentiality. Now we will look at preventing/detecting masquerade, modification and repudiation attacks using authentication techniques. Note that we consider digital signatures as a form of authentication.

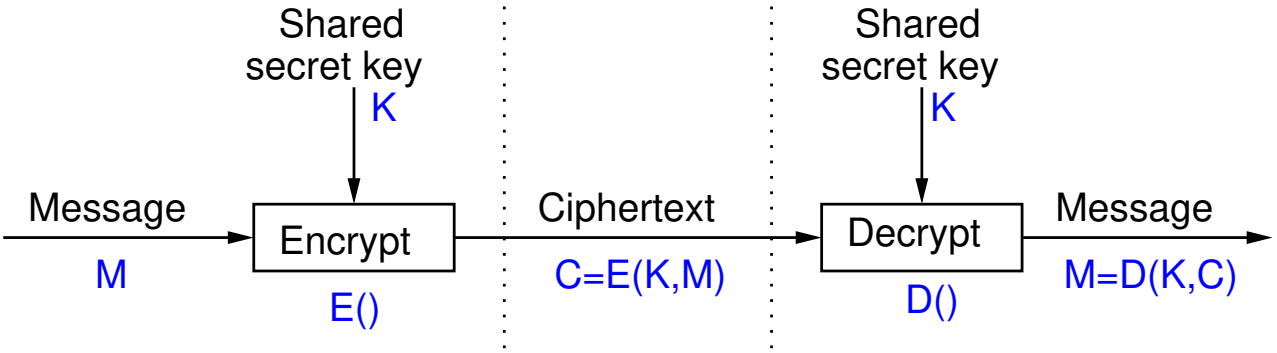
Aims of Authentication

- Receiver wants to verify:
  1. Contents of the message have not been modified (*data authentication*)
  2. Source of message is who they claim to be (*source authentication*)
- Different approaches available:
  - Symmetric Key Encryption
  - Hash Functions
  - Message Authentication Codes (MACs)
  - Public Key Encryption (i.e. Digital Signatures)

We will cover these different approaches in the following sections.

Contents

Symmetric Encryption for Authentication



The figure on slide 6 shows symmetric key encryption used for confidentiality. On the left is the sender A, and on the right is the receiver B. In the middle (between the dashed lines) is the information sent from A to B. Only B (and A) can recover the plaintext. However in some cases this also provides:

- Source Authentication: A is only other user with key; B knows it must have come from A
- Data Authentication: successfully decrypted implies data has not been modified

The source and data authentication assumes that the decryptor (B) can recognise that the result of the decryption, i.e. the output plaintext, is correct.

Recognising Correct Plaintext in English (question)

B receives ciphertext (supposedly from A, using shared secret key K):

`DPNFCTEJLYONCJAEZRCLASJTDQFY`

B decrypts with key K to obtain plaintext:

`SECURITYANDCRYPTOGRAPHYISFUN`

Was the plaintext encrypted with key K (and hence sent by A)? Is the ciphertext received the same as the ciphertext sent by A?

The typical answer for above is yes, the plaintext was sent by A and nothing has been modified. This is because the plaintext "makes sense". Our knowledge of most ciphers (using the English language) is that if the wrong key is used or the ciphertext has been modified, then decrypting will produce an output that does not make sense (not a combination of English words).

Recognising Correct Plaintext in English (question)

B receives ciphertext (supposedly from A, using shared secret key K):

`QEFPFPQEBTOLKDJBPPXDBPLOOVX`

B decrypts with key K to obtain plaintext:

`FTUEUEFTQIDAZSYQEEMSQEADDKM`

Was the plaintext encrypted with key K (and hence sent by A)? Is the ciphertext received the same as the ciphertext sent by A?

Based on the previous argument, the answer is no. Or more precise, either the plaintext was not sent by A, or the ciphertext was modified along the way. This is because the plaintext makes no sense, and we were expected it to do so.

Recognising Correct Plaintext in Binary (question)

B receives ciphertext (supposedly from A, using shared secret key K):

01101001101011010101101110000010

B decrypts with key K to obtain plaintext:

01011101000011010010101001011110

Was the plaintext encrypted with key K (and hence sent by A)? Is the ciphertext received the same as the ciphertext sent by A?

This is harder. We cannot make a decision without further understanding of the expected structure of the plaintext. What are the plaintext bits supposed to represent? A field in a packet header? A portion of a binary file? A random key? Without further information, the receiver does not know if the plaintext is correct or not. And therefore does not know if the ciphertext was sent by A and has not been modified.

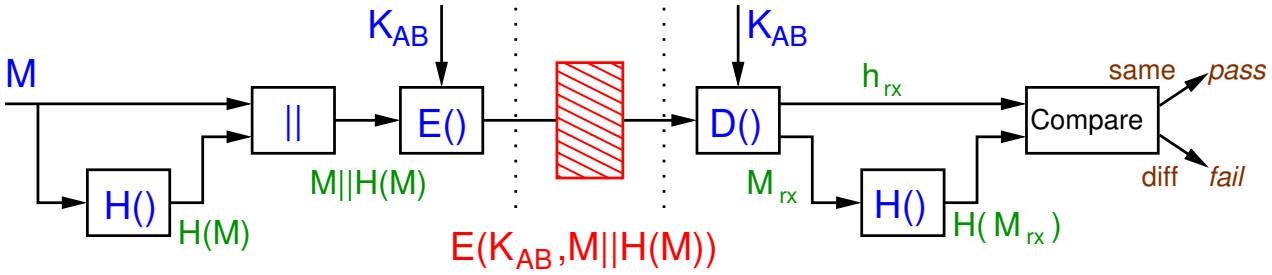
Recognising Correct Plaintext

- Many forms of information as plaintext can be recognised at correct
- However not all, and often not automatically
- Authentication should be possible without decryptor having to know context of the information being transferred
- Authentication purely via symmetric key encryption is insufficient
- Solutions:
  - Add structure to information, such as error detecting code
  - Use other forms of authentication, e.g. MAC

We will see some of the alternatives in the following sections.

Contents

Authentication by Hash and then Encrypt

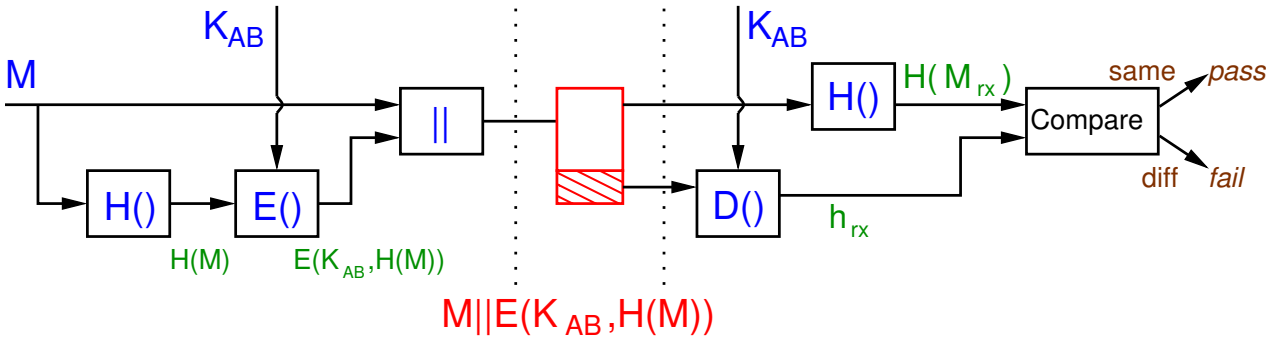


The figure on slide 12 shows a scheme where the hash function is used to add structure to the message. Again, user A and B are on the left and right, respectively. The inputs (message and secret key) and operations are shown in blue. The green values are used to refer to intermediate values. In the middle in red is the information sent from A to B.

At the receiver, the "received" message and hash are denoted with a subscript *rx*. In the normal case (no attack or error), the received values will be identical to the sent values, i.e. $M_{rx} = M$. However if an attack takes place, then it is possible the sent and received values differ.

When the receiver decrypts, they will be able to determine if the plaintext is correct by comparing the hash of the message component with the stored hash value. This is one method of addressing the problem of using just symmetric key encryption on its own for authentication. This scheme provides confidentiality of the message and authentication.

Authentication by Encrypting a Hash

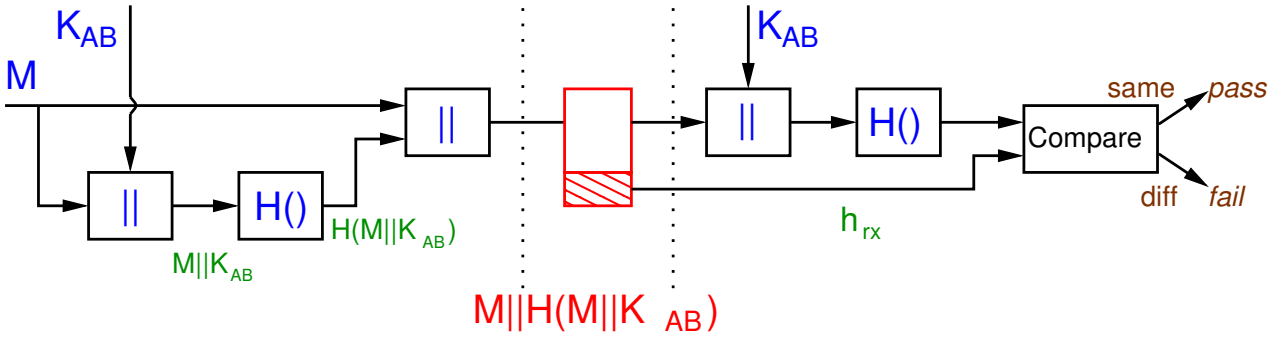


The figure on slide 13 shows a different scheme where only the hash value is encrypted. The receiver can verify that nothing has been changed. This scheme provides authentication, but does not attempt to provide confidentiality. This is useful in reducing any computation overhead when confidentiality is not required.

Attack of Authentication by Encrypting a Hash (exercise)

If a hash function did not have the Second Preimage Resistant property, then demonstrate an attack on the scheme in The figure on slide 13.

Authentication with Hash of a Shared Secret



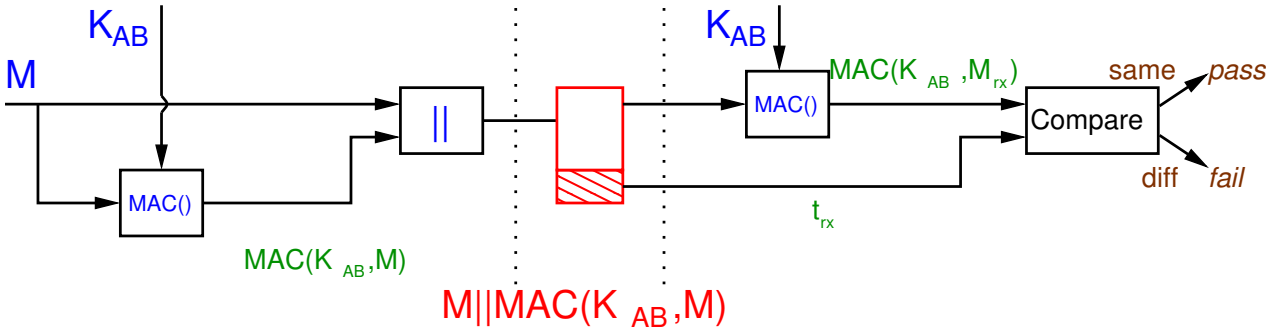
The figure on slide 15 shows a scheme the provides authentication, but without using any encryption. Avoiding encryption can be desirable in very resource constrained environments. S is a secret value shared by A and B. Concatenating the secret with the message, and then hashing the result, allows the receiver the verify the plaintext is correct, and keeps the secret confidential.

Attack of Authentication with Hash of Shared Secret (exercise)

If a hash function did not have the Preimage Resistant property, then demonstrate an attack on the scheme in The figure on slide 15.

Contents

Authentication with only MACs



The figure on slide 18 shows a scheme where authentication is provided using only a MAC. That is, encryption is not used.

Authentication using Encryption and a MAC

- Common to what both confidentiality and authentication (data integrity)
- MACs have advantage over hashes in that if encryption is defeated, then MAC still provides integrity
- But two keys must be managed: encryption key and MAC key
- Recommended algorithms used for encryption and MAC are independent
- Three general approaches (following definitions), referred to as authenticated encryption

Encrypt-then-MAC (definition)

The sender encrypts the message M with symmetric key encryption, then applies a MAC function on the ciphertext. The ciphertext and the tag are sent, as follows:

$$\mathrm{E}(K_1, M)||\mathrm{MAC}(K_2, \mathrm{E}(K_1, M))$$

Two independent keys, K_1 and K_2, are used.

MAC-then-Encrypt (definition)

The sender applies a MAC function on the plaintext, appends the result to the plaintext, and then encrypt both. The ciphertext is sent, as follows:

$$\mathrm{E}(K_1, M||\mathrm{MAC}(K_2, M))$$

Encrypt-and-MAC (definition)

The sender encrypts the plaintext, as well ass applying a MAC function on the plaintext, then combines the two results. The ciphertext joined with tag are sent, as follows:

$$\mathrm{E}(K_1, M)||\mathrm{MAC}(K_2, M)$$

Recommended Approach for Authenticated Encryption

- There are small but important trade-offs between encrypt-then-MAC, MAC-then-encrypt and encrypt-and-MAC
- Potential attacks on each, especially if a mistake in applying them
- Generally, encrypt-then-MAC is recommended, but are cases against it
- Some discussion of issues:
  - Chapter 9.6.5 of Handbook of Cryptography
  - Moxie Marlinspike
  - StackExchange
  - Section 1 and 2 of Authenticated Encryption by J Black
- Other authenticated encryption approaches incorporate authenticate into encryption algorithm
  - AES-GCM, AES-CCM, ChaCha20 and Poly1305

It is worth reading some of the discussion about the three approaches.

Contents

Digital Signatures

- Authentication has two aims:
  - Authenticate data: ensure data is not modified
  - Authenticate users: ensure data came from correct user
- Symmetric key crypto, MAC functions are used for authentication
  - But cannot prove which user created the data since two users have the same key
- Public key crypto for authentication
  - Can prove that data came from only 1 possible user, since only 1 user has the private key
- Digital signature
  - *Encrypt hash of message using private key of signer*

A digital signature has the same purpose of a handwritten signature: to prove that a document (or message or file) is approved by and originated from one particular person. If a message is signed, the signer cannot claim they did not sign it (since they are the only person that could create the signature). Similar, someone cannot pretend to be someone else, since they cannot create that other persons signature. Of course handwritten signatures are imprecise and sometimes forgeable. Digital signatures are much more secure, making it practically impossible for someone to forge a signature or modify a signed document without it being noticed.

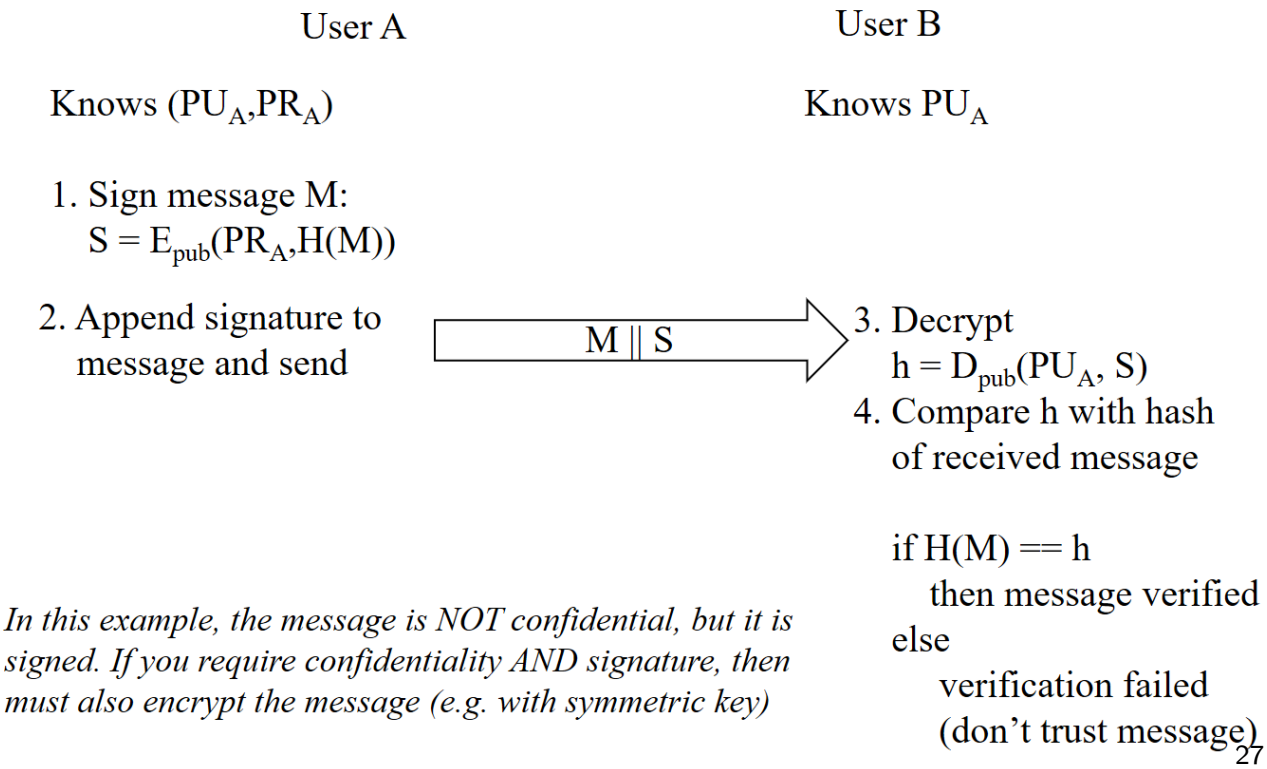
In practice, a digital signature of a message is created by first calculating a hash of that message, and then encrypting that hash value with the private key of the signer. The signature is then attached to the message.

The hash function is not necessary for security, but makes signatures practical (the signature is short fixed size, no matter how long the message is).

Digital Signatures in Practice

- User A has own key pair: (PU_A, PR_A)
- Signing
  - User A signs a message by encrypting hash of message with own private key: $S = E(PR_A, H(M))$
  - User attaches signature S to message M and sends to user B
- Verification
  - User B verifies a message by decrypting signature with signer's public key: $h = D(PU_A, S)$
  - User B then compares hash of received message, $H(M)$, with decrypted h; if identical, signature is verified

Digital Signature Example



In this example, the message is NOT confidential, but it is signed. If you require confidentiality AND signature, then must also encrypt the message (e.g. with symmetric key)